KU LEUVEN

Multi-tenancy in SaaS applications

Aggressive tenants

Tobias Lamote and Tomas Van Den Noortgate

Faculty of Computer Science

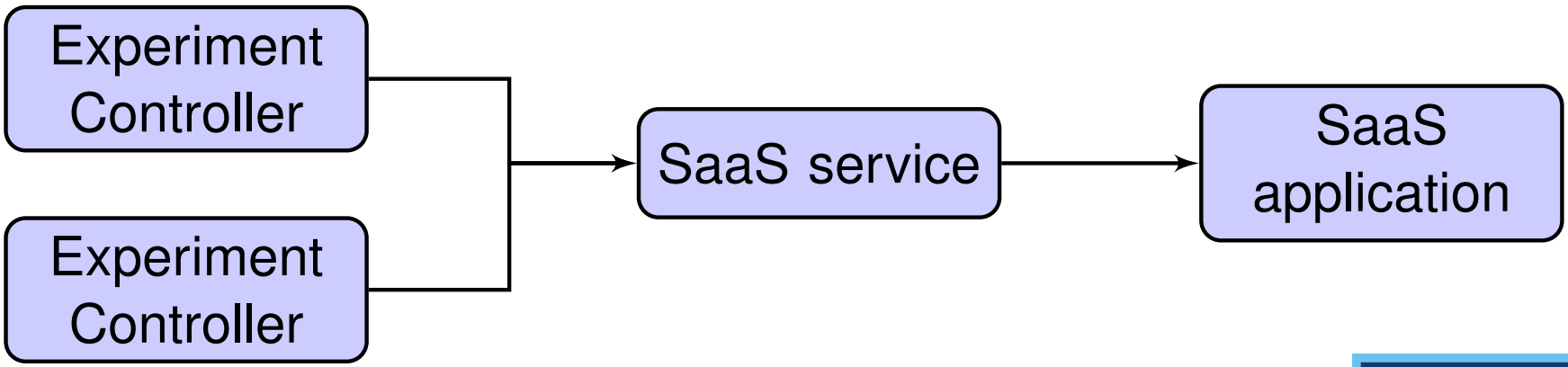
June 11, 2024

Multi-tenancy in Kubernetes

- What?
  - Host multiple users/applications on the same service
- Why?
  - More efficient use of resources
  - Less overhead for a new tenant
  - …
- How?
  - Service Level Agreement (SLA) for resources, latency,...
  - Quality of Service (QoS)

Introduction to SaaS application

- Purpose: simulate a real application
- Created in 2019
- Different configurations possible:
  - Memory
  - CPU
  - IO
- Used in conjunction with Experiment-controllers (on another node)



Aggressive tenants

Aggressive tenants

A tenant that does not respect the agreements and/or demands too much resources, which may impact the experience of other tenants.

- What should be done with ‘aggressive tenants’?
  - If not handled: impact on ”abiding” tenants
  - Deny access and/or deny requests
  - Give priority to higher level SLA’s
  - …

Problem statement

What if two tenants have the same SLA, and one of them is aggressive?

Experimental setup

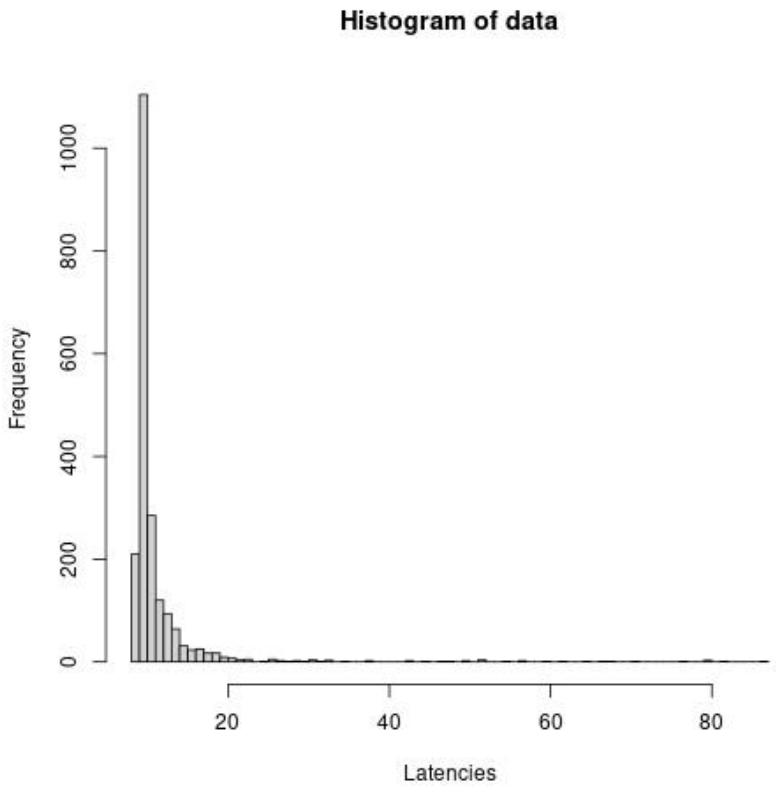
- *Assumption*: both tenants use same SLA
- Stressed in 2 ways: CPU and memory
- SLA consists of 20 requests/s
- Two kind of runs:
  - A first setup, where both tenants make 20 requests per second (RPS)
  - A second setup, where the normal tenant stays at 20 requests per second, but the aggressive tenant triples its requests (60 RPS)

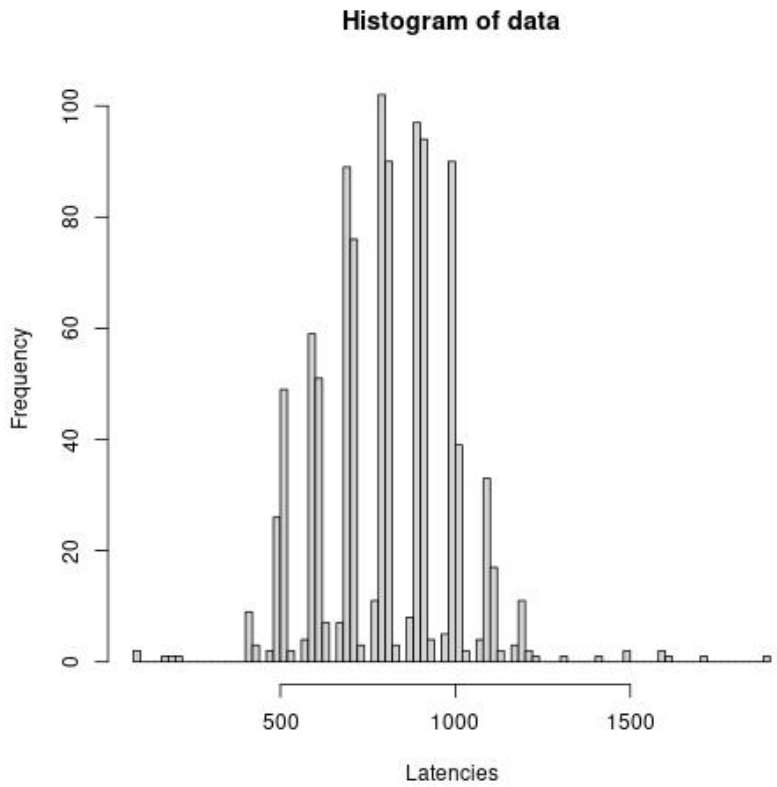
Hypothesis

Hypothesis

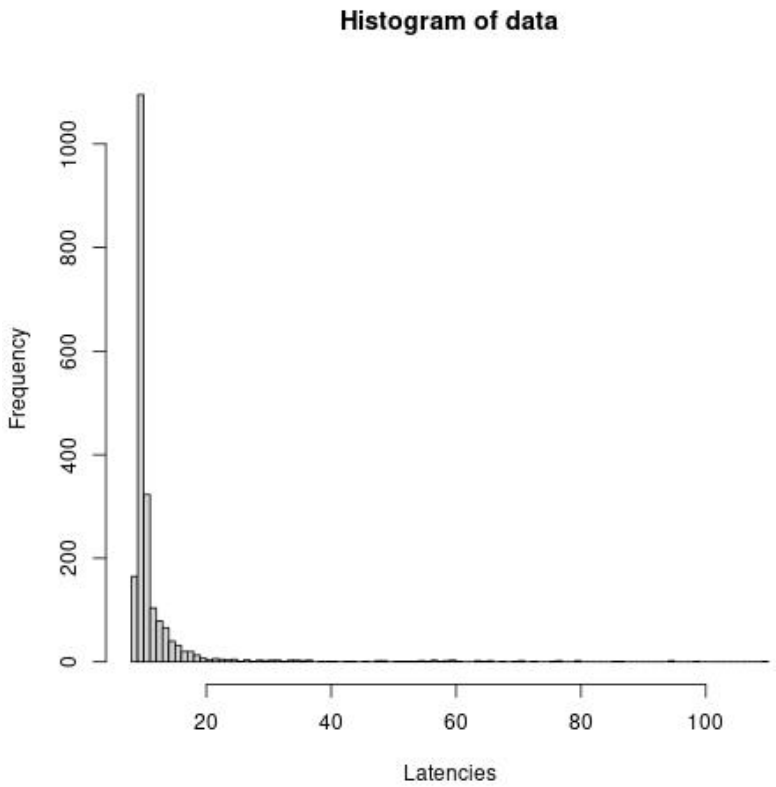
Due to the increase of requests from the aggressive tenant, the SaaS application will use relatively more resources for that tenant, past it's capabilities, thus resulting in worse performances for the *abiding* tenant.

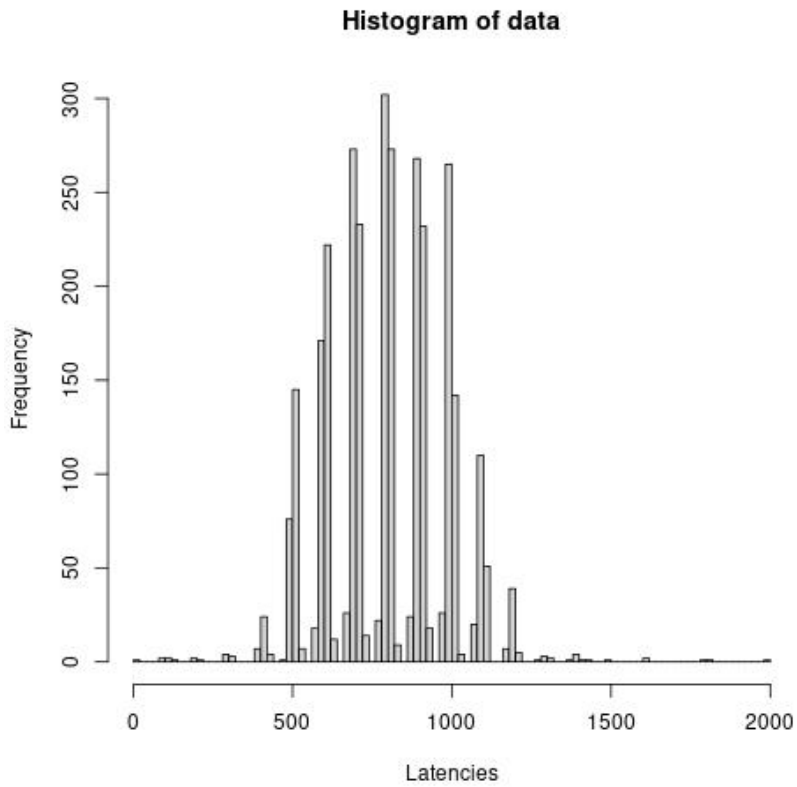
Abiding tenant, CPU bound, no policy





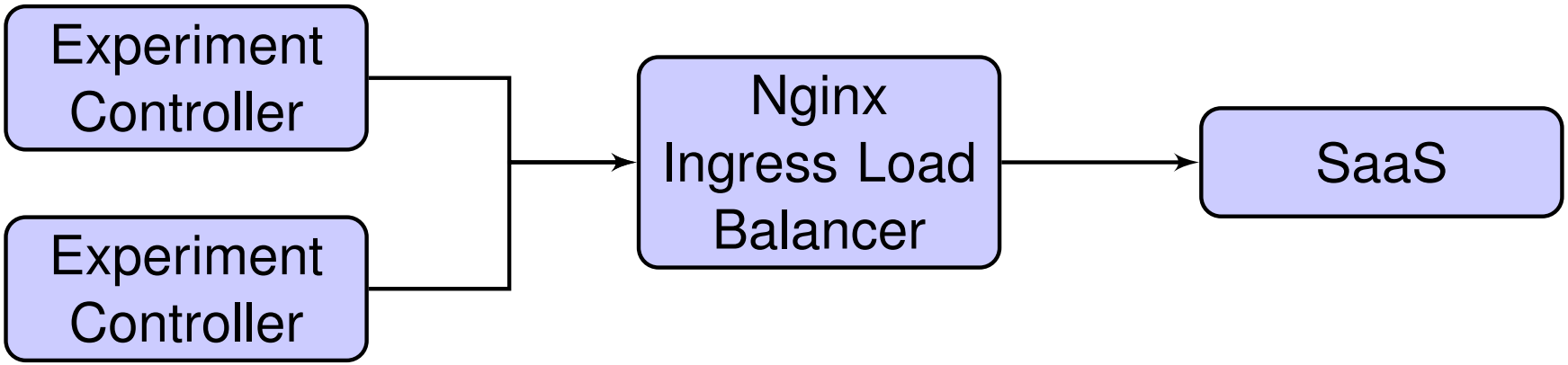
Aggressive tenant, CPU bound, no policy

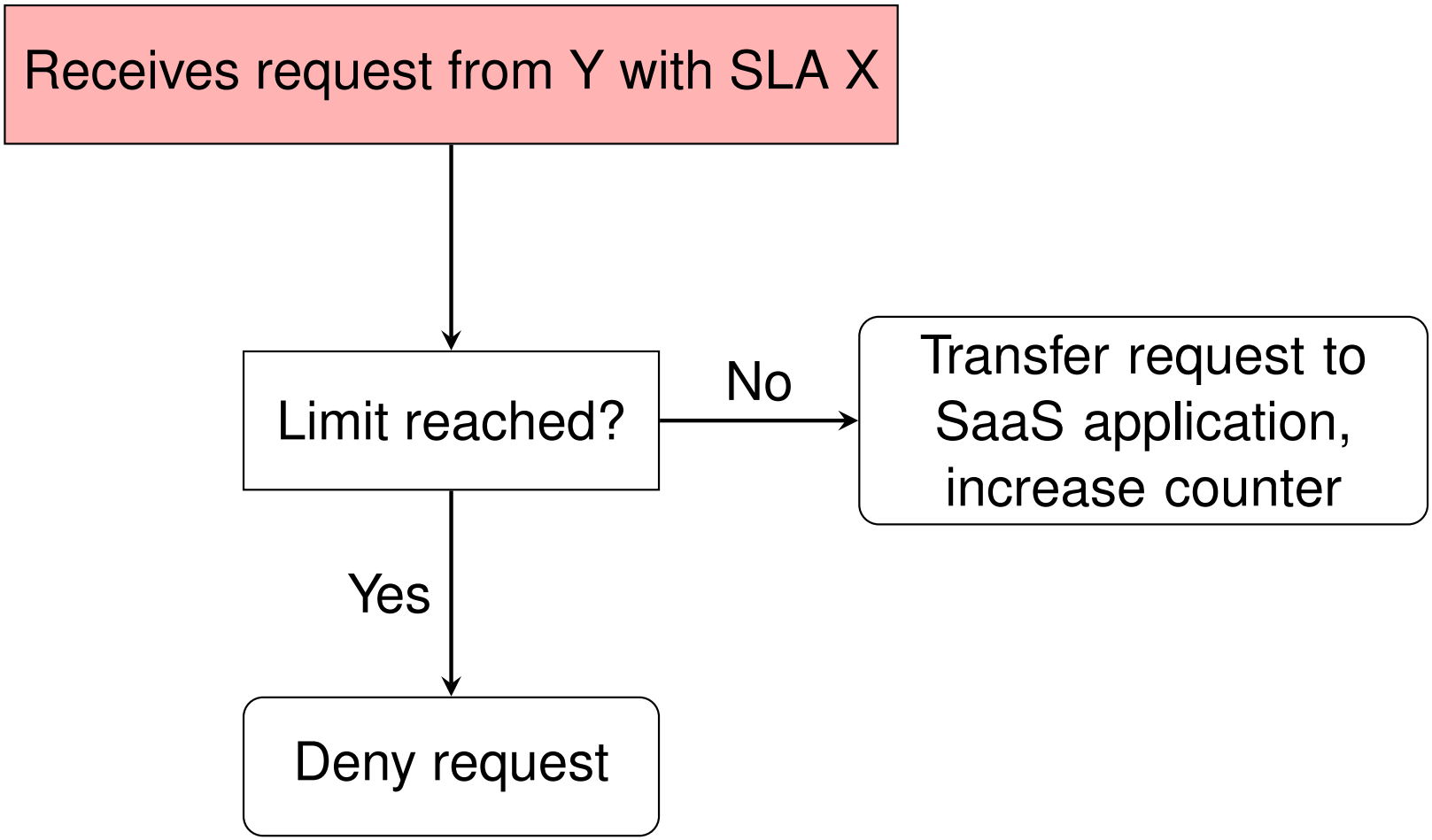




A first solution: Nginx Ingress Load Balancer

- Service that redirects and monitors http & https requests
- Allows specification of a “requests per second (RPS) limit” for all IPs
- Each IP has its own counter



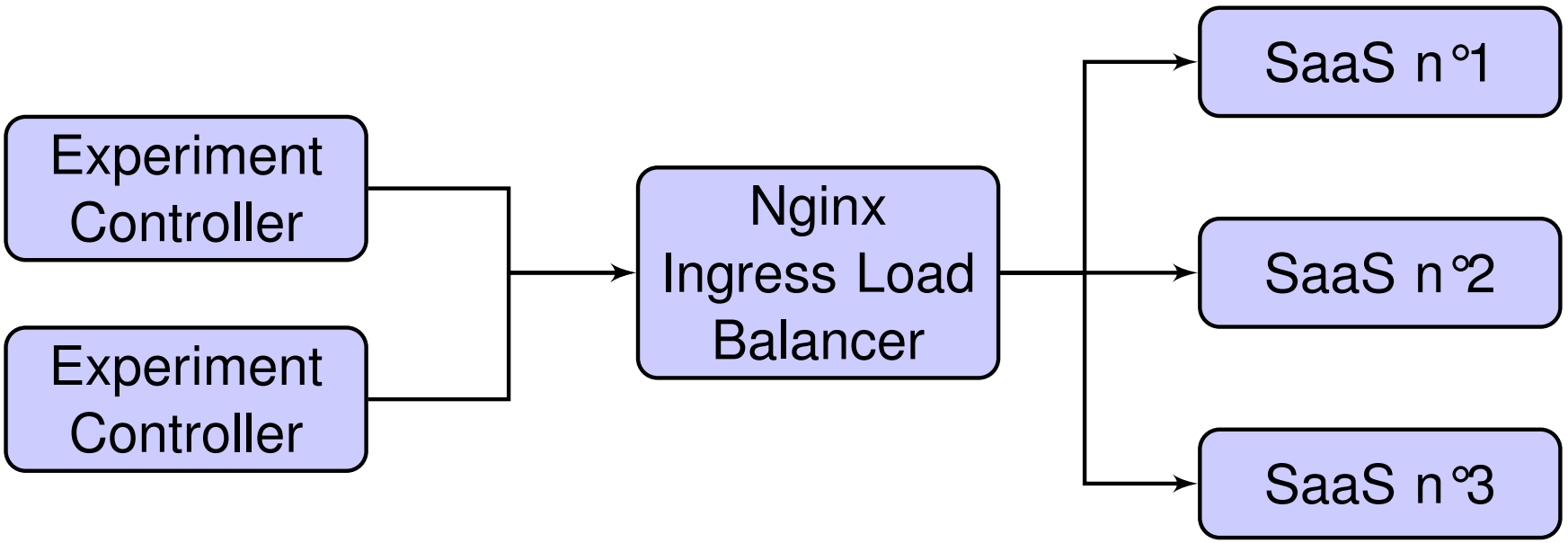

Receives request from Y with SLA X
Limit reached?
No
Transfer request to SaaS application, increase counter
Yes
Deny request

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: saas-ingress
  annotations:
    nginx.ingress.kubernetes.io/limit-rps: "20"
    nginx.ingress.kubernetes.io/limit-burst-multiplier: "1"
spec:
  rules:
  - host:
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: saas-app-service
            port:
              number: 80
  ingressClassName: nginx
```

Figure: Nginx Ingress Load Balancer configuration

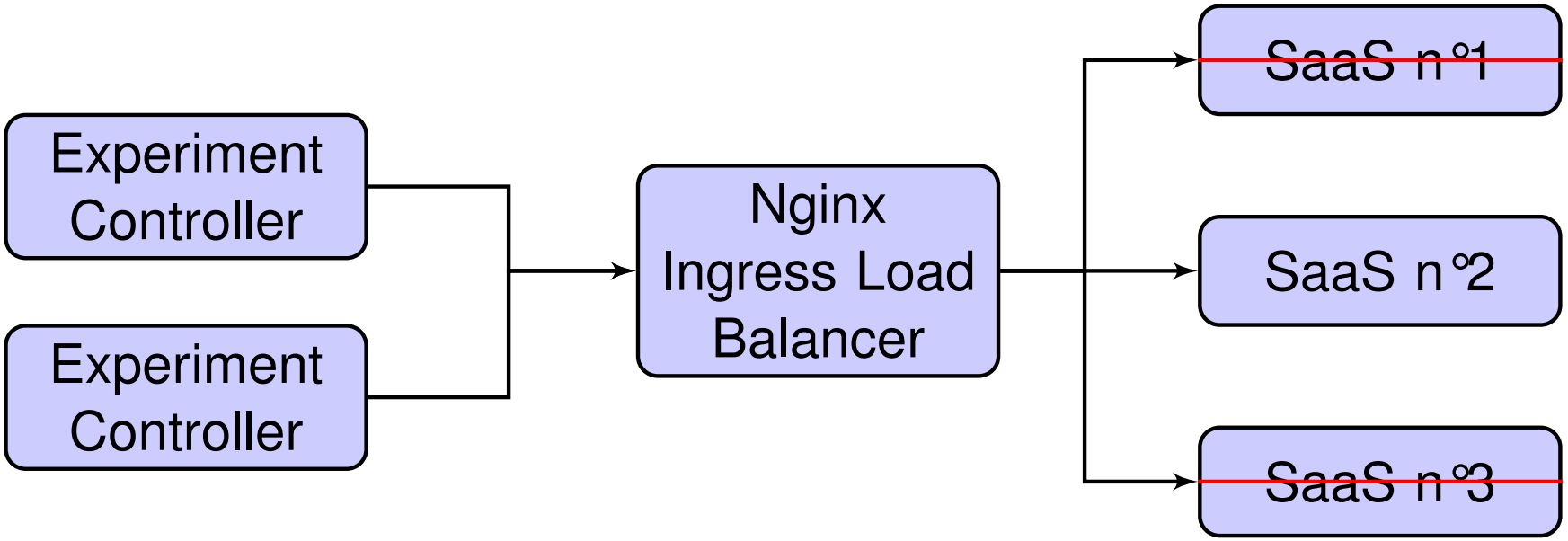
Note: separation of SLA's

- It is possible to separate the tenants from request (e.g. /request/1 would correspond to SLA 1)
- Request is redirected in function of SLA, one SaaS app on different cluster for each SLA



Note: separation of SLA's

- It is possible to separate the tenants from request (e.g. /request/1 would correspond to SLA 1)
- Request is redirected in function of SLA, one SaaS app on different cluster for each SLA
- We focused on tenants with same SLA

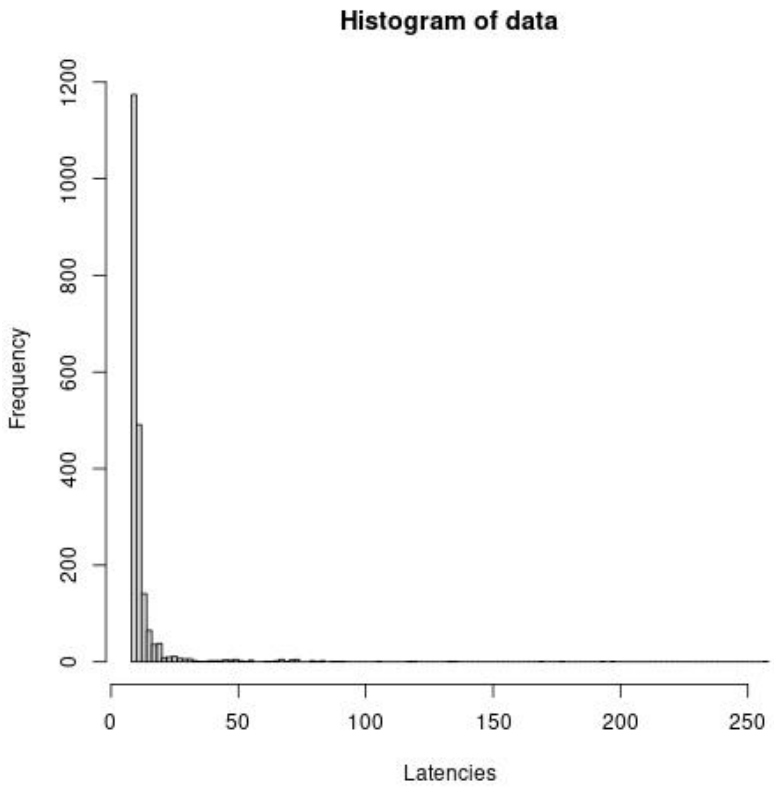


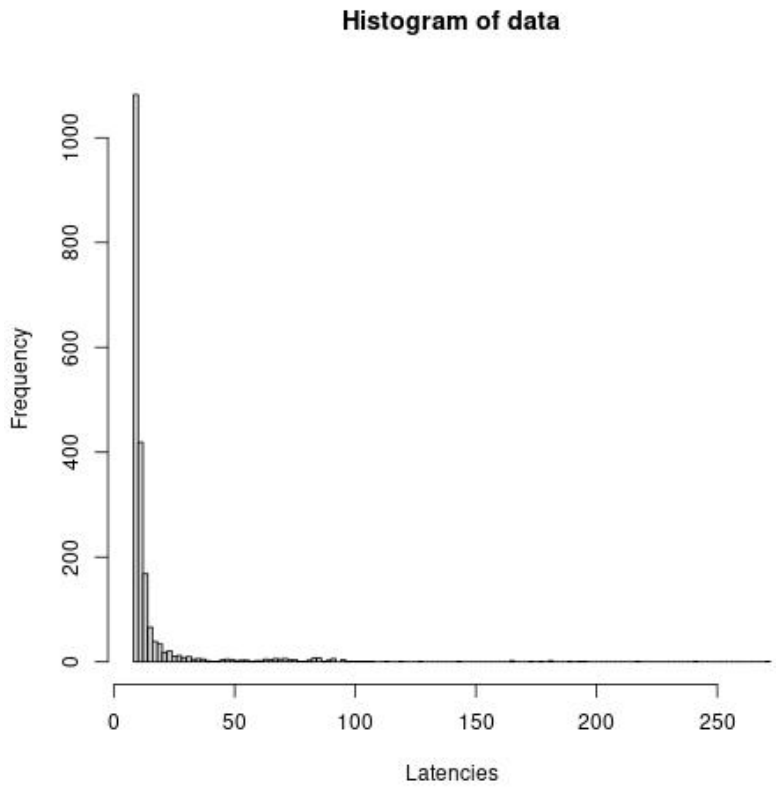
Hypothesis

Hypothesis

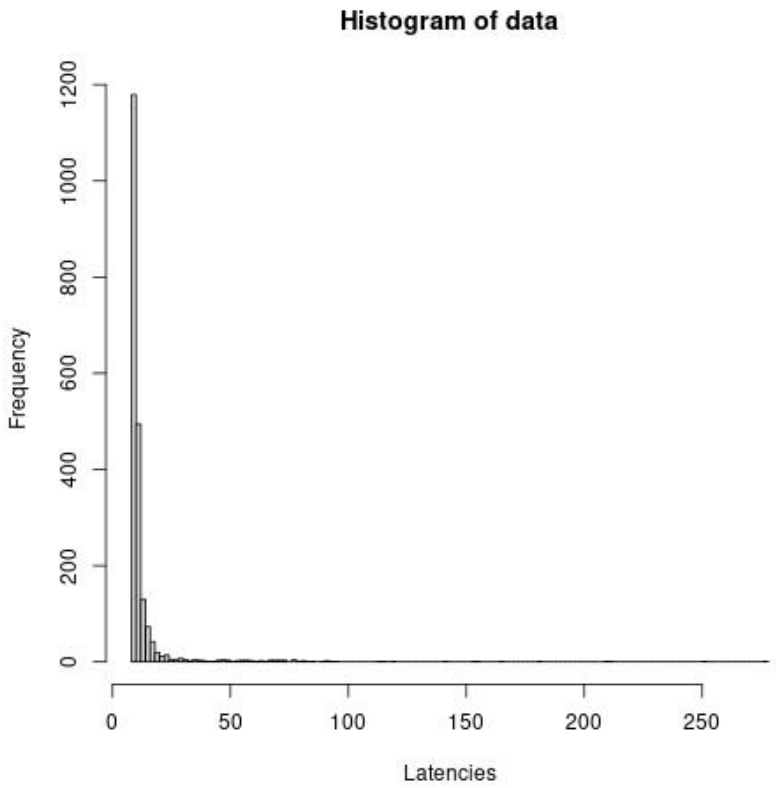
1. Due to the addition of an intermediate service (NILB), the average latency will increase.
2. The *aggressive* tenant will have no impact on the performance of the *abiding* tenant.

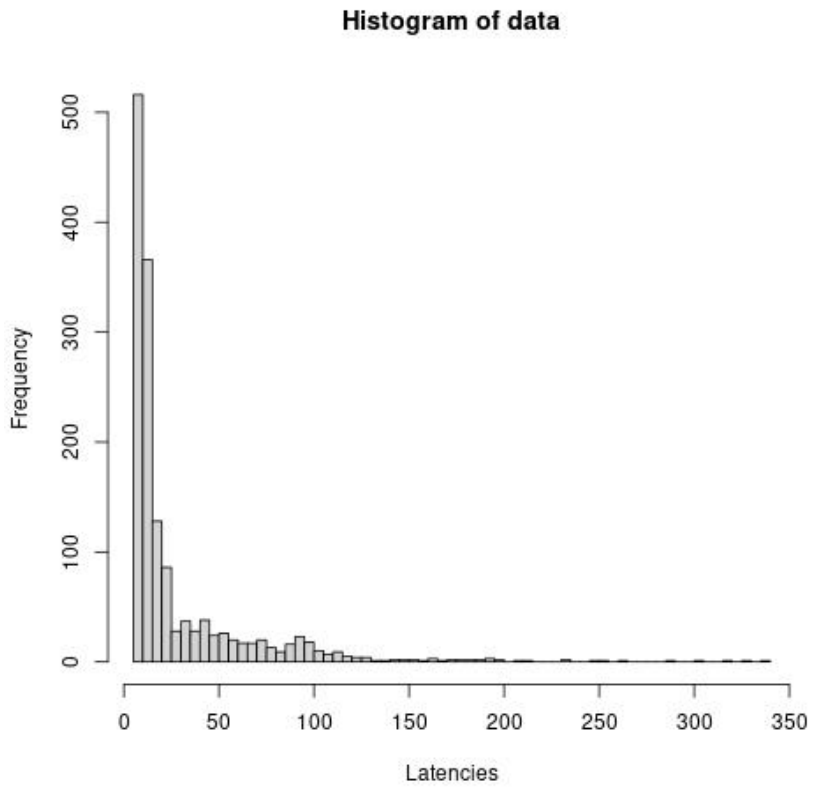
Abiding tenant, CPU bound, with Nginx





Aggressive tenant, CPU bound, with Nginx



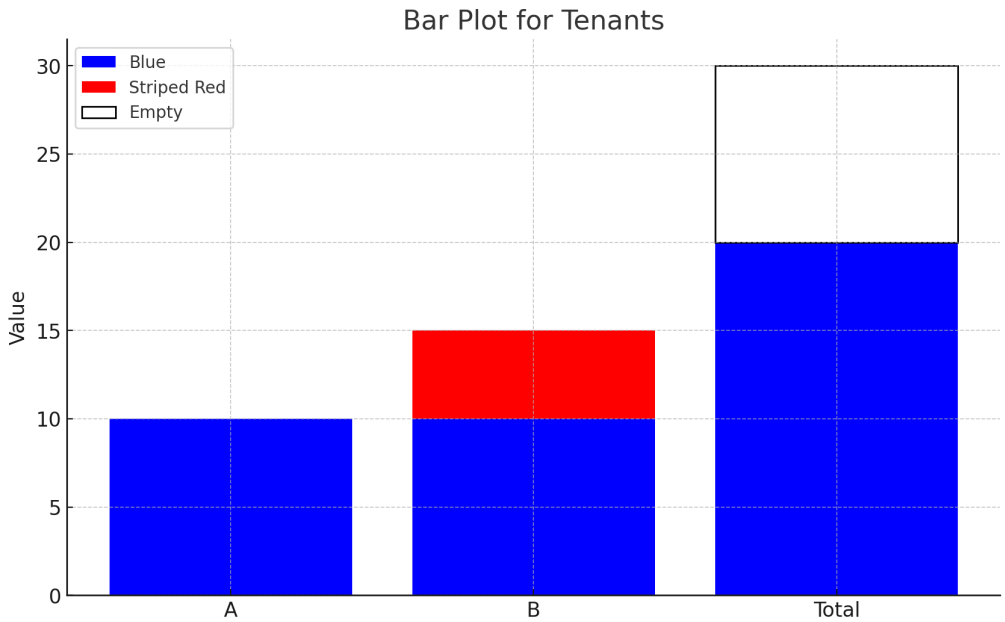


	Mean	SD	Median
Abiding tenant, CPU, no policy, inactive aggressive	11.27	6.4	9.66
Abiding tenant, CPU, no policy, active aggressive	803.10	192.8	800.89
Abiding tenant, CPU, Nginx, inactive aggressive	12.87	13.8	9.81
Abiding tenant, CPU, Nginx, active aggressive	15.70	20.6	9.88
Abiding tenant, MEM, no policy, inactive aggressive	6.90	3.1	6.12
Abiding tenant, MEM, no policy, active aggressive	110.51	89.3	100.81
Abiding tenant, MEM, Nginx, inactive aggressive	6.52	2.1	5.88
Abiding tenant, MEM, Nginx, active aggressive	7.22	3.9	6.33

	Mean	SD	Median
Aggressive tenant, CPU, no policy, inactive aggressive	12.03	8.9	9.71
Aggressive tenant, CPU, no policy, active aggressive	797.82	187.6	797.99
Aggressive tenant, CPU, Nginx, inactive aggressive	13.23	16.8	9.79
Aggressive tenant, CPU, Nginx, active aggressive	29.28	38.6	12.24
Aggressive tenant, MEM, no policy, inactive aggressive	6.75	1.9	6.16
Aggressive tenant, MEM, no policy, active aggressive	112.03	91.2	100.30
Aggressive tenant, MEM, Nginx, inactive aggressive	6.70	3.2	5.94
Aggressive tenant, MEM, Nginx, active aggressive	11.33	12.5	6.70

Going further . . .

- Need for different limits depending on SLA
- Non-optimal use of resources



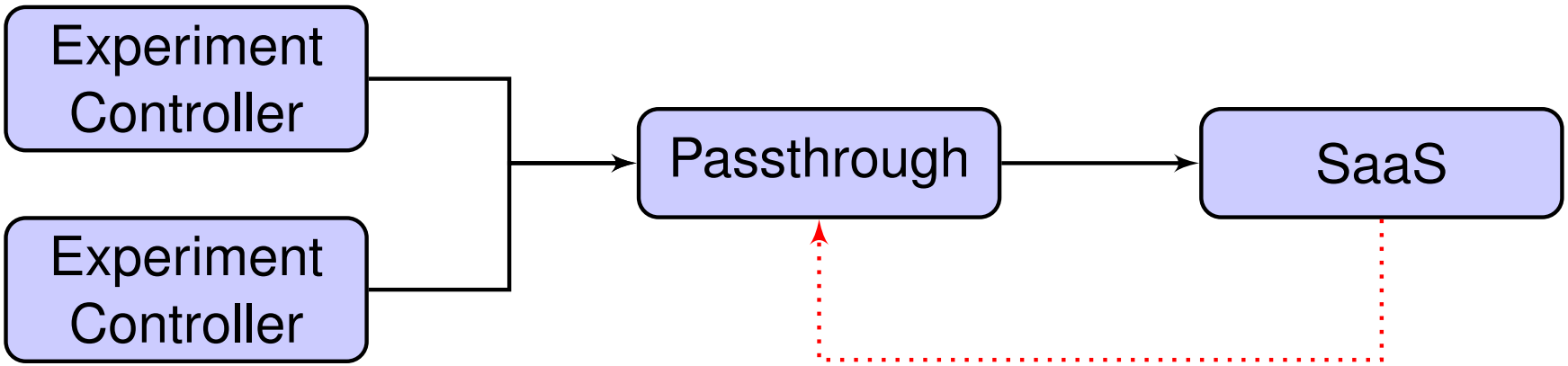
Going further . . .

```
metadata:
  name: whoami
  annotations:
    nginx.ingress.kubernetes.io/server-snippet: |
      geo $limit {
        default 5;
        10.244.205.193 10;
        10.244.151.1 1;
      }
      map $limit $limit_key {
        ~\b(10|5)\b $binary_remote_addr;
      }
      limit_req_zone $limit_key zone=req_limit_per_ip:10m rate=$limitr/s
spec:
  rules:
```

Figure: Server snippet to declare custom limits for each client.

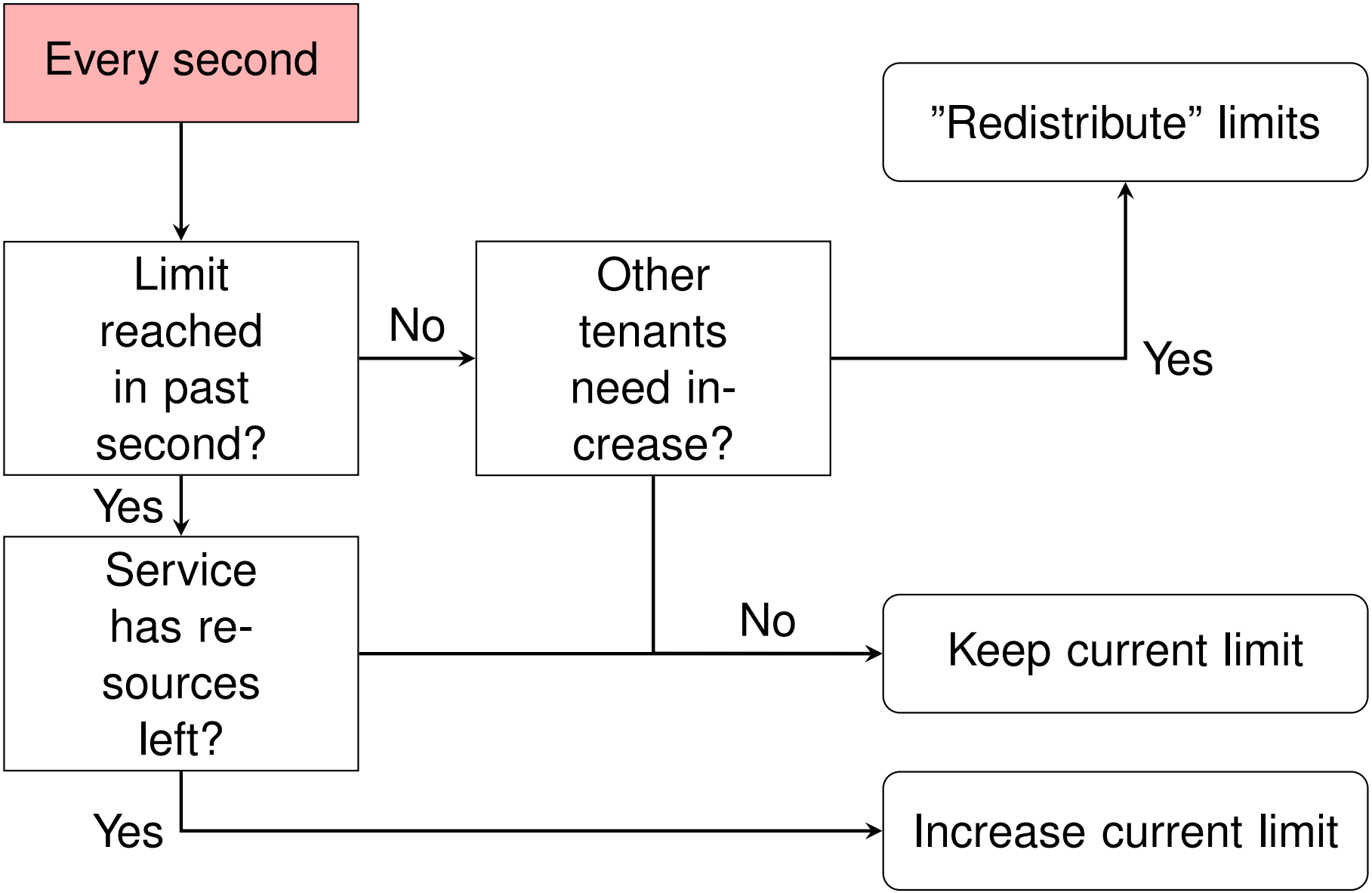
Passthrough

- Service that redirects and monitors requests, but with dynamic RPS
- Own Creation. Not functional, but proposed concept
- Based on Nginx Ingress Load Balancer
- Monitors resource availability of each SLA SaaS service



How does Passthrough work?

- Main idea similar to NILB:
  - Counts request per second (RPS) originating from each IP
  - If limit reached, deny request
- Has a limit for each IP
- Dynamically updates limits depending on usage of each SLA SaaS service
- Rest API for configuration


Every second
Limit reached in past second?
No
Other tenants need increase?
Yes
”Redistribute” limits
Yes
Service has resources left?
No
Keep current limit
Yes
Increase current limit

Key questions

Q: What effect does CPU topology have on the performance isolation?

- When there are more containers, you can pin them on certain cores
- Separate into normal and aggressive cores

Key questions

Q: What effect does topology management have on performance isolation?

- Influence pod scheduling decisions based on cluster topology
- Minimize interference and improve performance isolation by placing pods strategically
- Group aggressive tenants together, on separate nodes

Q: What about host privileged tenants (pods on same network)?

- Problematic, can circumvent our system by directly connecting to the SaaS service without going through NILB or Passthrough

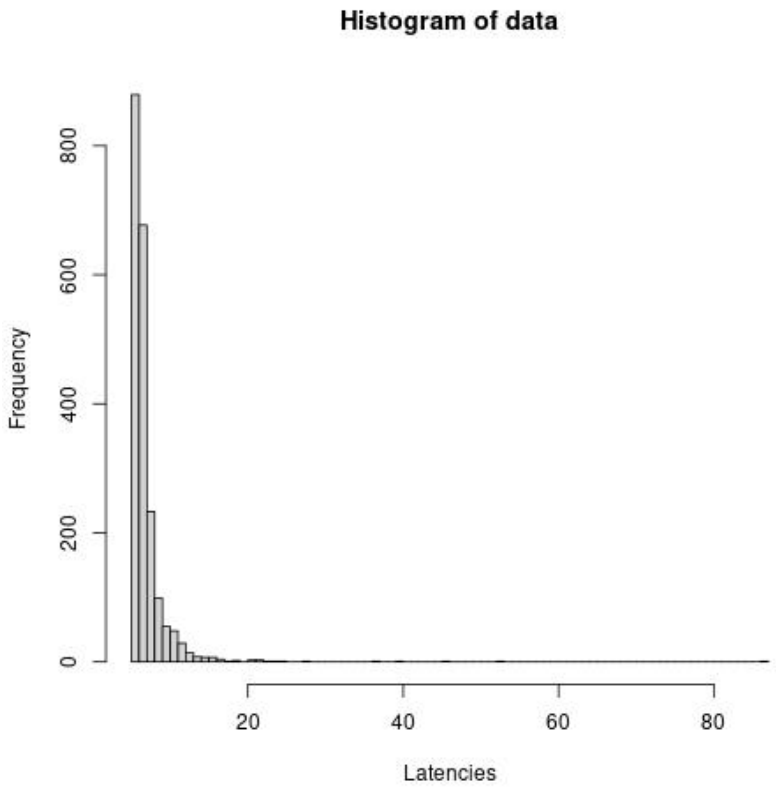
Discussion

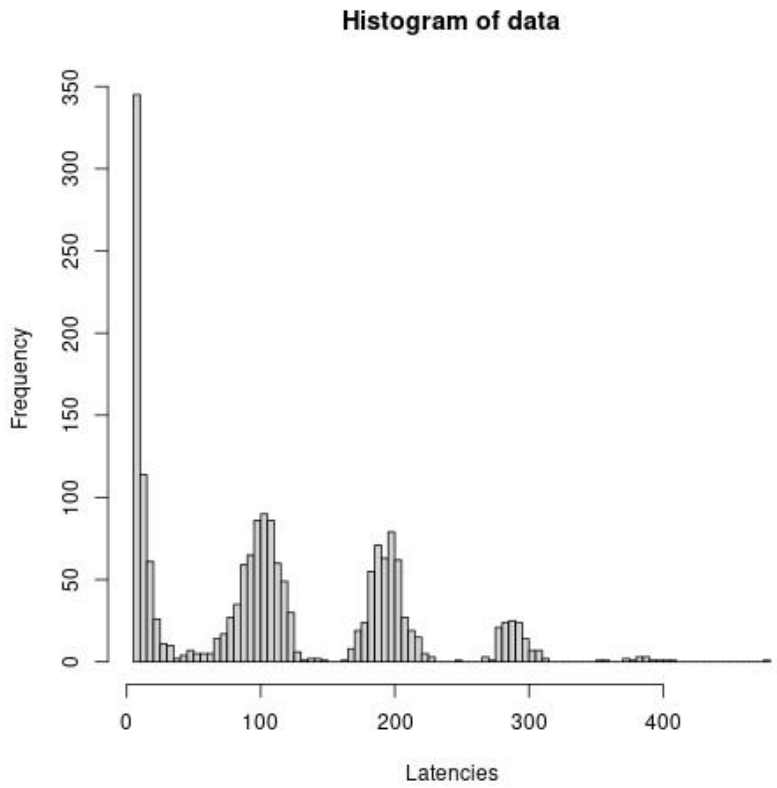
- Positive aspects
  - Encouraging results from Ingress Load Balancer
  - Theoretical improvement in resource usage from Passthrough
- But... Room for improvement
  - Outright denials are not ideal. Maybe a buffer?
  - The dynamic part can be improved using a monitoring service for the SaaS app
  - Slight performance cost due to middle man (increased delay)

Thank you for your attention!
Questions?

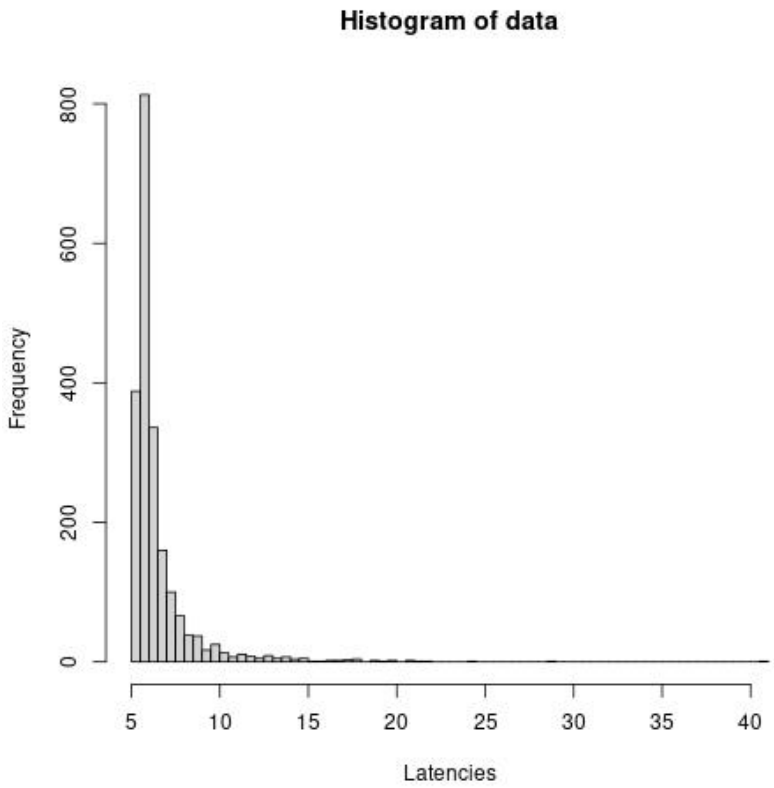
Topic	Tobias	Tomas
Research, first design, main ideas, . . .	50%	50%
Nginx, experiments	70%	30%
Passthrough, slides	30%	70%

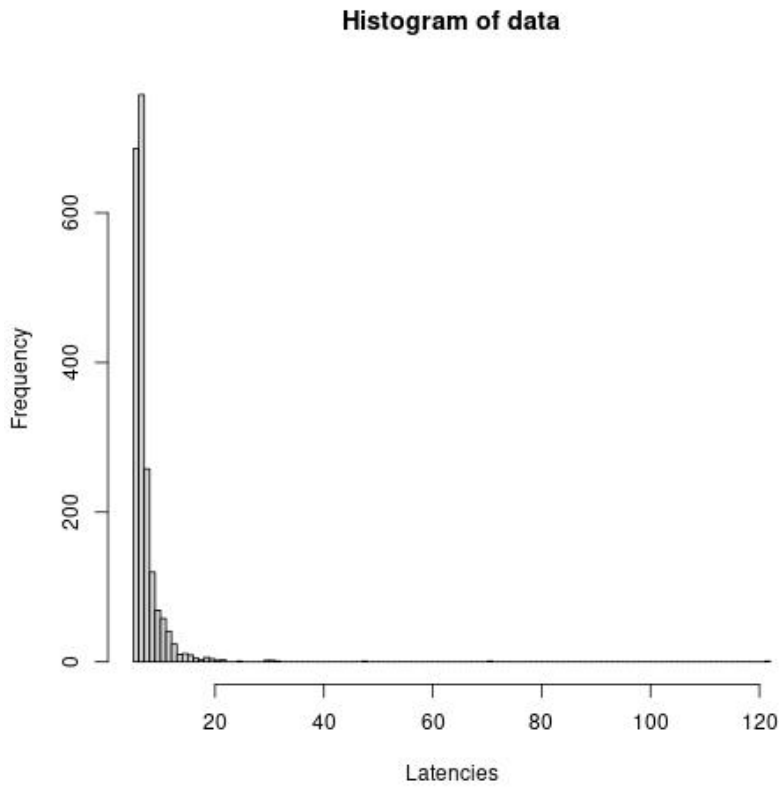
Abiding tenant, MEM bound, no policy



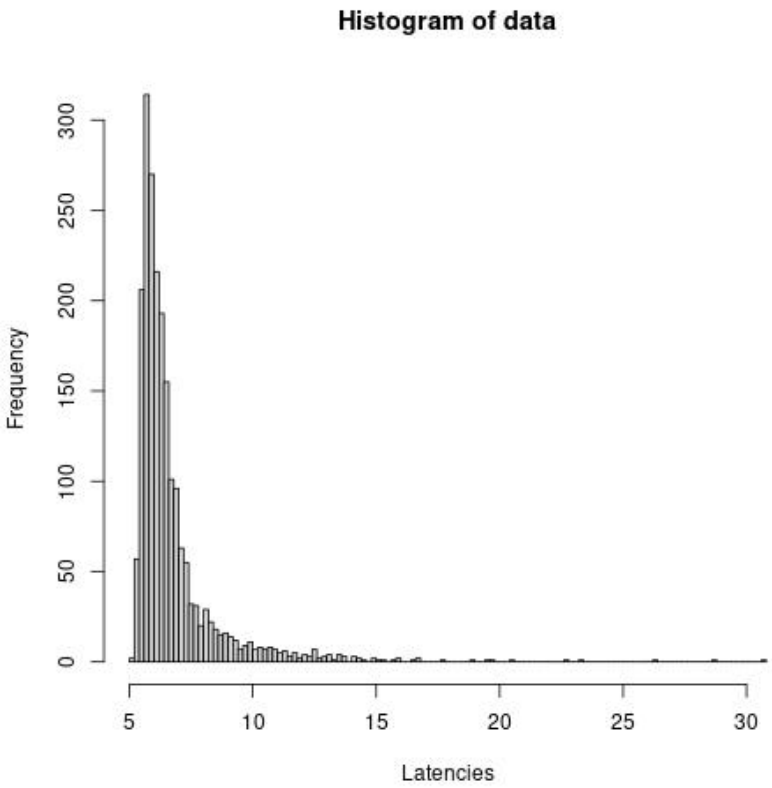


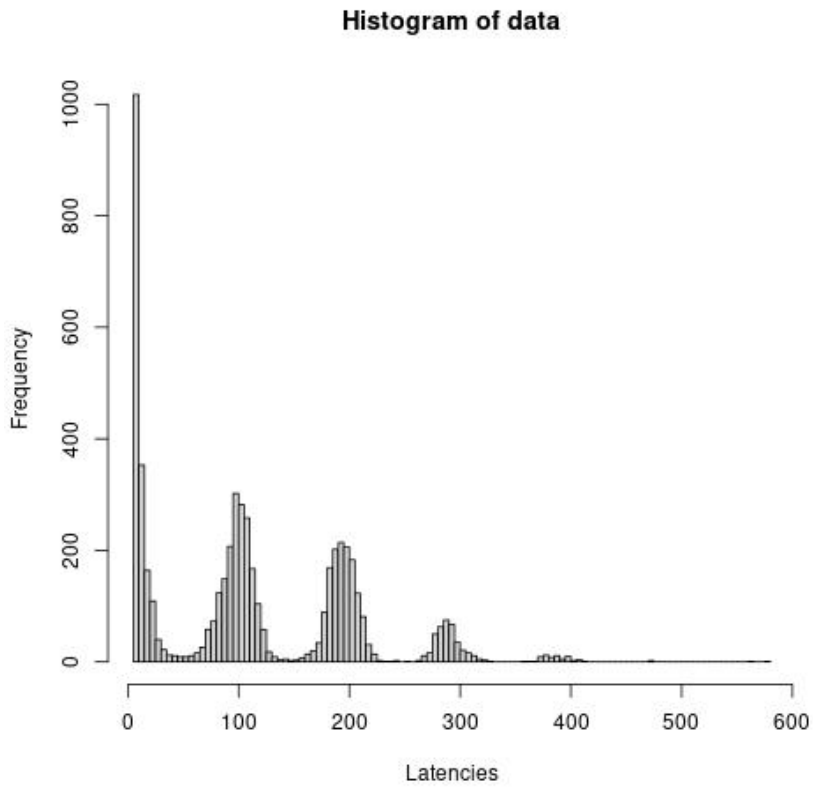
Abiding tenant, MEM bound, with Nginx





Aggressive tenant, MEM bound, no policy





Aggressive tenant, MEM bound, with Nginx